Est-il besoin de savoir programmer pour comprendre les fondements de l'informatique ou utiliser les logiciels ?

François Élie, Bastien Guerry, Dominique Lacroix, Philippe Lucaud, Charlie Nestel, Cécile Picard-Limpens, Thierry Viéville

Est-il besoin de savoir programmer pour comprendre les fondements de l'informatique ou utiliser les logiciels ?

François Élie, Bastien Guerry , Dominique Lacroix, Philippe Lucaud, Charlie Nestel, Cécile Picard-Limpens, Thierry Viéville

Ce texte collectif fait la synthèse d'échanges sur la liste de diffusion du groupe ITIC de l'association Enseignement Public et Informatique (EPI : www.epi.asso.fr), produits à la suite du partage des références : « Why everyone should learn programming » (2010), Daniel Shiffman & Mark Webster [1]*, et « Comment développer la culture en informatique : en l'enseignant dès le lycée » (2010), Antoine Petit* [2]*.*

Deux questionnements

Aurait-on d'un côté l'informatique en tant que science, fille ou soeur des mathématiques selon les nuances, ce qui légitimerait l'enseignement de l'*algorithmique* et du *traitement de l'information* dans les filières scientifiques, et de l'autre, la technologie en tant que science des objets techniques qui laisserait aux filières techniques le soin d'apprendre les *langages de programmation* et le *fonctionnement des machines* ? Les uns n'auraient nul besoin de savoir programmer, laissant le soin aux autres de s'acquitter de cette tâche de mise en oeuvre !

Une aristocratie et un tiers-état de l'informatique en quelque sorte [3].

Comme s'il n'existait pas d'ingénierie informatique, fondée pour une grande part sur des mathématiques appliquées...

Aurait-on d'un côté des utilisateurs éclairés des objets numériques, des artistes de l'usage informatique, qui devant la sophistication des ordinateurs et de leurs logiciels appropriés, suivraient un long apprentissage pragmatique pour en faire un usage précis et recherché, un usage qui leur est propre, et d'un autre, une minorité de créateurs de ces logiciels qui détiendraient alors le pouvoir planétaire de décider et d'imposer aux autres outils et objets à utiliser ?

Des utilisateurs à la merci des concepteurs en quelque sorte.

Comme si la notion de logiciel libre (qui permet d'accéder au code source de chaque logiciel, de l'étudier, de l'améliorer et de le repartager) n'existait pas...

Cinq clés pour y répondre

Cinq clés permettent de briser cette double idée reçue.

1. Apprendre à programmer permet de faire le lien entre

pratique et théorie

En informatique, la 1re machine est celle de Turing et c'est une simple abstraction mathématique [4] et de logique théorique ! La programmation est une affaire de langages formalisés [5], une notion algébrique ! Bref, ce que nous pouvions croire être du domaine technique est issu des plus élaborées des théories.

À l'inverse, la théorie de l'information [6] (qui permet de mesurer l'information comme une quantité physique en « bits ») et l'algorithmique [7] (qui permet de concevoir des logiciels) se tournent vers les objets techniques. Bref, technique et science ne s'opposent pas en informatique mais s'inter-mêlent avec bonheur (voir l'encart *Du terme « technologie »*) : bienvenue au 21e siècle.

Au lycée (voir l'encart *À propos de « B2I »*), l'algorithmique est maintenant un savoir bien identifié : on découvre les ingrédients des algorithmes [8] et le codage des objets numériques [9] en seconde générale, depuis 2008-2009, car il est enfin acquis que ces savoirs sont indispensables à l'ère du numérique. Mais comment s'incarnent ces savoirs abstraits ? Que sont les « travaux pratiques » de ces cours de science informatique ? Quels exercices pratiques permettent de s'approprier ces notions inédites ?

Réponse : *la programmation*. Programmer les algorithmes permet non seulement de vérifier que les savoirs sont compris, mais également d'entrevoir toutes les potentialités de l'informatique. La programmation favorise l'apprentissage par l'utilisation ce qui correspond bien à l'esprit humain, permet de découvrir une notion en la manipulant avant d'en abstraire la notion sous-jacente. On parle ici d'apprendre à implémenter des algorithmes quel que soit l'outil.

2. Tout outil logiciel qui devient programmable devient adaptable sans limite

Compositeur de musique ou photographe artistique ?

Musicien compositeur, il aura un logiciel pour écrire, dans le langage musical qui lui va, la séquence de notes qui une fois exécutée sur des instruments acoustiques ou numériques fera fonctionner sa création. La musique « d'avant » correspond au cas rudimentaire d'une simple séquence d'instructions de notes ou d'accords de durées (et autres paramètres) fixés. Pour aller plus loin (voir l'encart *Du compositeur numérique*), le compositeur voudra définir ses propres mécanismes pour créer les briques de base de sa musique et ses propres mécanismes pour les assembler, bref, il voudra passer de l'*énumération* des notes et accords de sa musique à la *programmation* de l'évènement musical souhaité.

Artiste photographe, il aura un logiciel pour arranger ses images en indiquant à la souris où et quoi changer, mais s'il conçoit des effets photographiques qui vont au-delà de ce que la palette de fonctions du logiciel peut proposer, il devra lui même *programmer* les opérations de transformation de son image.

La programmation est un nouveau médium qui reste au service de la création, et ne supplantera jamais la main humaine.

Ce qui est remarquable en informatique, c'est que dès qu'un logiciel dispose des ingrédients des algorithmes en plus de ses fonctions prédéfinies, il permet alors de programmer tous les algorithmes possibles, c'est à dire d'implémenter tout ce qu'un ordinateur peut calculer de manière mécanique : il devient universel, il n'y aucune transformation numérique, aucun processus qui ne puisse pas être réalisé. C'est ce résultat immense que le milieu du 20^{e} siècle a offert à l'humanité, nous permettant de comprendre ce qu'est l'intelligence mécanique. Apprendre à programmer, c'est au niveau des principes apprendre à tout faire.

3. Apprendre à programmer permet le partage au sein du monde numérique

Au niveau pratique, c'est ne plus être limité ou bridé par tel ou tel logiciel.

Ainsi, comme le présente l'association Ars Industrialis [10] : *« Sur Internet, chacun peut le constater à tout moment, il n'y a pas d'un côté des producteurs, et de l'autre des consommateurs : la technologie numérique ouvre un espace réticulé de contributeurs, qui développent et partagent des savoirs, et qui forment ce que nous avons appelé un milieu associé... Ce partage, qui reconstitue des processus de sublimation, et qui reconstruit en cela une économie productrice de désir, d'engagement et de responsabilités individuelles et collectives socialement articulées selon de nouvelles formes de sociabilités, ouvre un espace... ».*

Le phénomène *hackerspaces* est exemplaire à ce niveau. Un hackerspace est un lieu où des personnes d'intérêt commun touchant l'informatique, les nouvelles technologies, l'art digital ou électronique, peuvent se rencontrer, échanger, et/ou collaborer [11]. Le hackerspace peut être considéré comme un laboratoire mettant à disposition des ressources matérielles (machines, outils et locaux), mais aussi organisant des ateliers, permettant alors aux *hackers* d'y partager des ressources et des savoirs, dans le but d'élaborer et de faire des choses. Les hackerspaces font usage de logiciels libres et de médias alternatifs, mais aussi contribuent à leur développement. Pour ce qui est de l'étendue internationale du mouvement, somme toute non négligeable, voir [12] et aussi [13]. Cette démarche se retrouve également dans les FabLabs [14], concourt à l'idée de ré-appropriation des connaissances et au partage de savoirs et pratiques.

Bien sûr, celui qui ne sait pas ou ne sait pas encore programmer n'est pas exclu du monde numérique, il peut faire des choses profondes en utilisant des logiciels dédiés, il est juste privé d'une *partie* des possibles de l'informatique. Même s'il n'est pas possible d'y consacrer beaucoup de temps, il faut déjà en donner la curiosité.

L'important, c'est d'avoir le choix.

4. Apprendre les bases de la programmation est un apprentissage facile

C'est une centaine d'heures de travail. Bien moins que pour apprendre à lire ou écrire, à peu près comme apprendre à conduire. On parle ici de pouvoir adapter le code dans une page web dynamique de son site, réaliser un petit automatisme

dans sa maison pour piloter des appareils « domotisés », ajouter à Facebook une application qui permet d'aller plus loin dans l'usage d'un tel logiciel, offrir un cadeau personnel sous forme de mini-jeux à un proche, adapter un logiciel libre à sa façon.

Cela signifie apprendre un langage usuel (comme PHP ou JAVA, selon ce que l'on souhaite en faire) et le pratiquer au sein d'une structure associative, scolaire ou professionnelle. Apprendre les bases algorithmiques, puis s'entraîner sur des exemples « inutiles » mais instructifs, pour partager ensuite avec d'autres les morceaux de logiciels qui permettront de faire ce qui nous sied.

Cela veut dire que le responsable du site web de votre petite entreprise n'est pas handicapé pour ajouter un formulaire ou un petit calcul à la page web du site. Que le médecin généraliste qui a une méthode répétitive éprouvée pour trier et ranger ses documents administratifs ne va pas « cliquer la même chose à chaque fois » mais lancer une seule commande pour tout réitérer. Que celui qui aura tâtonné des heures pour faire marcher l'installation d'un nouveau produit ne va pas devoir en plus écrire tout un mode d'emploi pour permettre au suivant d'arriver à la solution, mais simplement programmer la liste des commandes pertinentes que le suivant exécutera immédiatement.

Au-delà, devenir un professionnel du logiciel nécessite plusieurs années de travail, bien évidemment. Apprendre à programmer au niveau citoyen permet de mieux comprendre ces métiers. Dans sa maison, savoir « faire de l'électricité » permet à la fois de ne pas déranger le professionnel pour une broutille, de ne surtout pas croire pouvoir bricoler soi-même ce qui implique des courants forts, et de s'assurer que le professionnel vous vend un service pertinent, sans profiter de votre crédulité. Apprendre à programmer a les mêmes vertus quotidiennes.

5. Apprendre à maîtriser la programmation est aussi un enjeu de pouvoir

Il y a deux manières de « maîtriser » la technique de la programmation : (i) la maîtriser en lui attribuant une fin (s'en servir), (ii) la maîtriser par rapport aux besoins qu'elle produit et qui renforce sa domination. C'est donc un enjeu sociétal (voir l'encart *Maîtriser la programmation et enjeu sociétal*).

(i) La programmation aide-t-elle à attribuer une fin ? Nous l'avons discuté précédemment. Elle permet de choisir sa finalité. Et donc de dominer son environnement numérique plus efficacement au moyen de la technique... Apprendre la programmation permet donc aussi de comprendre comment se libérer de ceux qui veulent nous imposer leur fin par la technique. Parce qu'il y a deux manières de programmer : viser la programmation... des hommes, ou viser la libération des hommes.

Certains qui ont appris à programmer trouveront donc de leur intérêt personnel que la programmation ne soit pas enseignée... D'autres verront comme très nécessaire que chacun soit instruit de cela. De ce point de vue, le fait que certains ne veuillent pas partager ce qu'ils ont reçu d'enseignement de la programmation, rend encore plus nécessaire qu'elle soit enseignée! Et l'enjeu est *international* : un pays qui aurait imposé à la majorité du monde entier une unique base logicielle (un

système d'exploitation [15]), disposerait d'une suprématie inouïe. Ouvrir la porte, donc, faire entrevoir ce qu'il est possible de faire.

(ii) La programmation aide-t-elle par ailleurs à maîtriser sa technique ? Et à éviter la séduction de la technique (de l'usage) ? Comme cela peut sembler paradoxal, si l'on en juge par le *geek* ou le *no-life* passionné à ce point de programmation qu'il ne fait « que » programmer! Comment croire que celui qui a prévu de regarder un film et qui passe une heure à réinstaller les *codecs*, convertit, recolle le son et re-synchronise les sous-titres, et qui ne regardera pas le film, est l'homme libre ? Le suprême aliéné n'est-il pas l'homme du *tuning* ? C'est là qu'il faut distinguer deux programmations. Celle qui se fait passer pour telle et qui n'est que configuration. Et la vrai programmation, qui est algorithmique.

Voilà donc une autre nécessité de l'enseignement de la programmation que de savoir faire ce distinguo.

Une nouvelle vision en émergence

Notre argumentaire permet de sortir de la fausse antinomie Science/Outil qui réserverait aux uns l'enseignement de la science informatique, quel que soit son nom (voir l'encart *Des noms pour dire « informatique »*), et aux autres ses aspects techniques : pour une culture scientifique, mais également et conjointement pour une culture technologique. Et le formidable mouvement du *logiciel libre* va dans le sens de la « maîtrise » au sens développé ici et implique un apprentissage général et citoyen de la programmation [21].

François Élie <francois.elie@adullact.org>,
Bastien Guerry <bastienguerry@googlemail.com>,
Dominique Lacroix <dl@panamo.eu>,
Philippe Lucaud <philucpro@yahoo.fr>,
Charlie Nestel <cnestel@free.fr>,
Cécile Picard-Limpens <ccl.picard@gmail.com>,
Thierry Viéville <thierry.vieville@inria.fr>

Références

[1] « Why everyone should learn programming » (2010), Daniel Shiffman & Mark Webster.
http://flowingdata.com/2010/10/28/why-everyone-should-learn-programming

[2] « Comment développer la culture en informatique : en l'enseignant dès le lycée » (2010), Antoine Petit.
http://interstices.info/enseignement

[3] *Ces préjugés qui nous encombrent* (2009), Gilles Dowek, éd. Du Pommier.

[4] Wikipédia « Machine de Turing »
http://fr.wikipedia.org/wiki/Machine_de_turing

[5] Wikipédia « Théorie des langages »
http://fr.wikipedia.org/wiki/Théorie_des_langages

[6] Wikipédia « Théorie de l'information »
http://fr.wikipedia.org/wiki/Théorie_de_l'information

[7] Wikipédia « Algorithmique »
http://fr.wikipedia.org/wiki/Algorithmique

[8] « Les ingrédients des algorithmes » (2010), Gilles Dowek, Thierry Viéville, Jean-Pierre Archambault, Emmanuel Baccelli, Benjamin Wack.
http://interstices.info/jcms/c_43821/les-ingredients-des-algorithmes

[9] « Tout a un reflet numérique » (2010), Gilles Dowek, Thierry Viéville, Jean-Pierre Archambault, Emmanuel Baccelli, Benjamin Wack.
http://interstices.info/jcms/c_43823/tout-a-un-reflet-numerique

[10] Ars Industrialis, « Manifeste 2010 » §3.
http://arsindustrialis.org/manifeste-2010

[11] Wikipedia « Hackerspace »
http://en.wikipedia.org/wiki/Hackerspace
http://fr.wikipedia.org/wiki/Hacklab

[12] « List of Hacker Spaces »
http://hackerspaces.org/wiki/List_of_Hacker_Spaces

[13] « La prochaine révolution, faites la vous même » (2010), Jean-Marc Manach.
http://www.lemonde.fr/technologies/article/2010/11/05/la-prochaine-revolution-faites-la-vous-meme_1436212_651865.html#xtor=AL-32280258

[14] « Faire émerger et connecter des FabLabs en France »
http://fing.org/?Le-Fab-Lab-lieu-d-artisanat&lang=fr

[15] Wikipédia « Système d'exploitation »
http://fr.wikipedia.org/wiki/Système_d'exploitation

[16] « B2i®, C2i® ... et autres attestations informatique et Internet »
http://www.educnet.education.fr/dossier/b2ic2i

[17] « Un chemin initiatique vers l'abstraction » (2010), Gilles Dowek
http://www.epi.asso.fr/revue/articles/a1009g.htm

[18] « La question de la technique » (1954), Martin Heidegger.
http://agora.qc.ca/textes/heidegger.html
et Wikipédia « Technologie »
http://fr.wikipedia.org/wiki/Technologie

[19] Labo Sony de musique
http://www.csl.sony.fr/research/music

[20] « Toutes les libertés dépendent des libertés informatique » (2003), Richard Stallman.
http://ecrans.blogs.liberation.fr/ecrans/2006/06/tribune_ecran_n.html

[21] « L'Informatique : Science, Techniques et Outils » (1998), Bernard Lang.
http://www.datcha.net/ecrits/ailf

[22] Les machines à voter électroniques en débat... (2007), Chantal Enguehard, Isabelle Maugis.
http://interstices.info/vote-electronique

[23] « Hacker and painters » (2003), Paul Graham.
http://www.paulgraham.com/hp.html

À propos de « B2I »

Au primaire et collège se fait l'apprentissage des usages en informatique. L'objectif est que chaque enfant de France ait accès à l'informatique, apprenne à utiliser les logiciels de communication et de bureautique et surtout à user correctement d'internet et de son application principale : le web. L'apprentissage de l'utilisation des logiciels (à travers les B2I et C2I [16]) ne se discute plus, sinon lorsque les ressources humaines ou logistiques font défaut ou que le contenu réel des acquis n'est pas clarifié.

Le B2I devrait idéalement former à l'apprentissage du web et de ses pièges, mais aussi d'un traitement de texte et d'un tableur, ainsi que savoir manipuler le système d'exploitation de son ordinateur, ce qui permet d'apprendre à aller chercher sur les forums la solution aux problèmes rencontrés, ceci avec une évaluation réelle des acquis et des cours spécifiques (au moins 1h par semaine semble nécessaire). La réalité n'est pas si idyllique [17] ! (Retour au texte)

Du terme parfois ambigu de « technologie »

Jusqu'à la révolution industrielle c'est principalement le mot « art », qui est employé pour désigner une manière de faire, une façon de, une maîtrise de, pour être progressivement remplacé par « technique » qui va désigner un savoir-faire, tandis que l'art va prendre le sens de poésie, de « création ». Le mot technologie émerge également au XVIII^{e} siècle, d'abord en Allemagne, dans la lignée des encyclopédistes, pour désigner le discours sur les techniques. Les planches de métier de l'*Encyclopédie de Diderot et d'Alembert* préfigurent la géométrie descriptive de Monge, qui lors de la Révolution française soutint la première chaire de Technologie et fonda également l'École Polytechnique. Avec le « technologue »,

s'opère donc une convergence entre les sciences et les techniques. Il en est de même aux États-Unis où, par exemple, à sa création en 1861, le MIT (*Massachusetts Institute of Technology*) empruntera son nom, et aussi de nombreuses orientations pédagogiques, à la convergence qui s'opérait à l'aube de la révolution industrielle entre les arts (*technè*) et la science (*logos*) [18]. Une convergence jusqu'alors compromise par l'impossible articulation des savoirs scientifiques fragmentés et des arts nécessairement enfermés dans une tradition.

À cette définition, s'oppose un autre usage du terme « technologie » : il désigne par là l'ensemble des connaissances et des pratiques mises en oeuvre pour offrir à des usagers des produits ou des services. La technologie fait donc dans cette acception intervenir, à côté des processus de transformation, des éléments relevant de la conception des produits et services, des attentes des usagers, de la fiabilité qu'ils recherchent, des prix qu'ils sont prêts à payer, du volume des marchés, des caractéristiques des matériaux disponibles, des compétences des travailleurs concernés.

Nous nous référons bien entendu au premier sens dans ce document. (Retour au texte)

Du compositeur numérique

Comme avec un traitement de texte, le musicien peut corriger, copier et coller, vérifier les notes de sa matière musicale. Mieux encore, il peut définir des objets musicaux plus complexes (des accords de musiques, des rythmes temporels, des sonorités mariant graves et aigues, des nuances de tons et de dynamiques...) qu'il peut ensuite manipuler comme des « super-notes », ces objets ayant tous les paramètres possibles à ajuster, pour lui permettre de faire exploser les possibles de sa création. Il peut encore spécifier non plus sa musique comme une simple séquence de notes, mais la définir comme en Jazz moderne, sous forme d'un assemblage conditionnel ou itératif de séquences de sons, pour que l'oeuvre ne soit plus simplement un simple chapelet musical, mais puisse vivre avec des variantes sans bornes. Il peut aussi confier au hasard la déclinaison de certaines variantes de sa création, pour choisir à moindre effort la réalisation qu'il reconnaît avoir voulu sans avoir pu la produire lui même, ou bien laisser son oeuvre évoluer au gré de ses possibles. Mais qu'est il en train de faire ce compositeur de musique du 21^{e} siècle ? Il est en train de *programmer* son oeuvre musicale !!! Chaque note est une instruction élémentaire, chaque objet musical plus évolué une fonction de ces instructions de base, les ingrédients pour les assembler un algorithme. La musique « d'avant » correspond au cas rudimentaire d'une simple séquence d'instructions de notes ou d'accords de durées.

Ici nous distinguons bien composer, interpréter et improviser. Même si le fait d'improviser et d'interpréter peut être parfois assimilé à une composition instantanée, la composition garde une spécificité : elle est un ensemble de notes (rythmes, harmonies) que l'on peut rejouer différemment mais qui garde une identité propre. Même si le hasard amène une idée, l'esprit est toujours là pour valider ou non son occurrence. Ce n'est donc pas le créateur du logiciel qui est le compositeur, mais celui qui crée la matière musicale avec. De plus, si une boite à rythme peut être certes programmée de telle manière qu'il est quasi impossible de distinguer si c'est un batteur qui joue ou une machine, il manquera « juste » un petit quelque chose à l'auditeur : la poésie, la beauté, l'émotion dues au fait que c'est un être humain qui lui a offert cela. Ça peut être beau mais ça manque d'humanité.

Le compositeur de musique, avec la programmation, peut externaliser de manière inouïe ses intentions et idées, et construire sa musique qui n'existera au final que jouée par l'interprète. Par exemple, au laboratoire de musique Sony de Paris [19], M. Pachet, a mis au point un algorithme qui « répond » en improvisation de jazz au phrasé de l'artiste. Et cet écho numérique lui permet de doubler sa puissance de création. (Retour au texte)

Apprendre à maîtriser la programmation et enjeu sociétal

Karl Marx dans *Contribution à la critique de l'économie politique* relatait déjà en 1856 qu'une machine-outil pouvait se décomposer en trois aspects : la partie commande, la force motrice et la partie opérative. Cette dernière ne relève-t-elle pas précisément de cette révolution informatique ? Et là, effectivement, en filigrane de cette révolution numérique se pose la question de la maîtrise, du pouvoir et Richard Stallman de nous dire : *« Toutes les libertés dépendent de la liberté informatique, elle n'est pas plus importante que les autres libertés fondamentales mais, au fur et à mesure que les pratiques de la vie basculent sur l'ordinateur, on en aura besoin pour maintenir les autres libertés. Profitant de la faiblesse de la démocratie contemporaine, les grandes entreprises sont en train de prendre le contrôle de l'État, ce sont elles qui contrôlent les lois, pas les citoyens. Ça a commencé avec le* Digital Millenium Copyright Act *aux États-Unis, puis elles ont imposé des directives européennes dans leur intérêt. »* [20]

Par ailleurs, tant que ne seront pas créés, tant dans les filières technologiques que dans l'enseignement professionnel, des enseignements liés à l'ingénierie informatique, il n'y aura pas équité face à la société numérique. Il faut contribuer au développement d'un enseignement de l'informatique dite science, tout en soutenant un

enseignement de l'informatique orienté métiers et/ou sciences de l'ingénieur qui permettrait de réintroduire les options nouvelles technologies appliquées qui existaient au collège en quatrième, réintroduire l'enseignement de notions informatiques dans les programmes de technologie du collège, réintroduire des ateliers de travaux dirigés dédiés à l'apprentissage de logiciels et de la programmation.

Entre temps, notre pays recrute de plus en plus ses compétences informatiques à l'international, donc devient dépendant en matière de savoir [21].

Autrement dit, apprendre l'informatique au sens défini ici est un levier à la fois de liberté et d'égalité républicaine. L'enseignement de la programmation permet de doublement « maîtriser » la technique : de comprendre les fins qui sont les siennes (telles qu'elles sont définies par d'autres), de comprendre les fins que chacun peut lui assigner et de comprendre comment échapper au mouvement dans lequel risquent de nous enfermer les « innovations » informatiques – celles, par exemple, qui concernent les machines de vote [22]. (Retour au texte)

Des noms pour dire « informatique »

Tandis que les « sciences informatiques » sont devenues « sciences et technologies de l'information et de la communication » (STIC) avant de passer par « sciences du numérique » (à ne pas confondre avec les « sciences numériques » qui comme la médecine numérique ou l'ingéniérie numérique utilisent les sciences du numérique pour travailler de manière numérique !), Paul Graham [23] nous explique la cause du problème « *Je n'ai jamais aimé le terme* science informatique *(*computer science*, en anglais). La principale raison est que ce n'est pas une chose unique en soi. L'informatique est un fourre-tout de plusieurs activités humaines assemblées par un accident de l'histoire. À une extrémité il y a des gens qui sont vraiment mathématiciens, mais appeler ce qu'ils font de l'informatique permet juste d'obtenir des subventions* [publiques]*. Au centre, vous avez les personnes travaillant sur quelque chose comme l'histoire naturelle des ordinateurs – l'étude du comportement des algorithmes pour le routage des données par réseaux, par exemple. Et puis à l'autre extrême vous avez les hackers, qui essaient d'écrire un logiciel intéressant, et pour qui les ordinateurs ne sont que moyens d'expression, comme le béton pour les architectes ou la peinture pour les peintres. C'est comme si mathématiciens, physiciens, architectes devaient* [être amalgamés]*. »*

Et bien il y a tout de même un dénominateur commun à cette mosaïque de disciplines informatiques : le besoin d'avoir compris la programmation. (Retour au texte)

Association EPI
Décembre 2010